Stack Problem

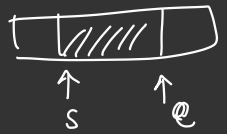
Q Array containing N elements. sum of maximum element from every subarray

[1, 4, 3]

[1] — 1
[4] — 4
[3] — 3
[1,4] — 4
→ ~~[1,3]~~ Not a subarray
[4,3] — 4
[1,4,3] — 4

20

N elements



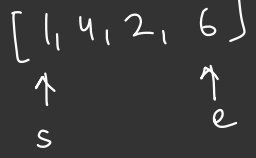
- for(s → start 0 — n-1) ~ O(N²) Subarrays
  - for(e → end s — n-1) {
    - sum = sum + Max of (a[s] .. a[e])

loop O(N)

O(N³)
O(N²)

3
3

Use carry Forward O(1)

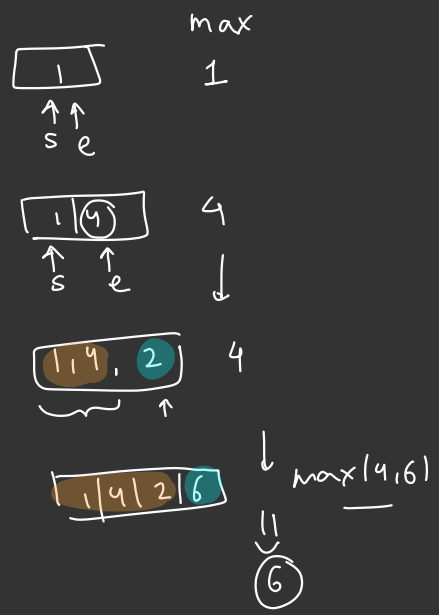


⇒ max can be computed using CF

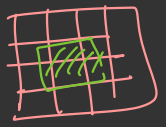
[4] – 4

[4, 2] – 4

[4, 2, 6] – 6



2D arrays



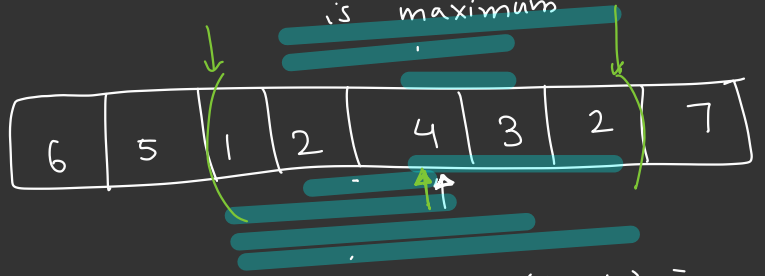
sum of every submatrix

Idea: for each element of the find its total contribution in sum

↓

No of subarrays in which that element is maximum

Example

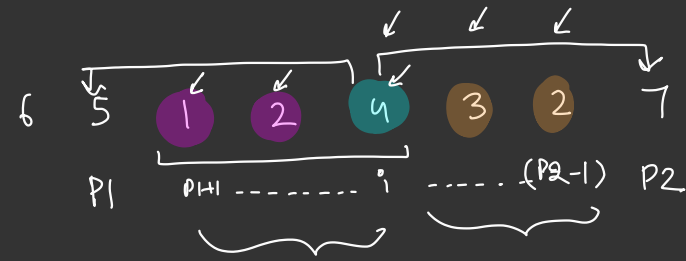


EC of 4 = 9 × 4 = 36 (efficiently)

→ [4] -
→ (2,4) -
→ (2,4,3) -
→ (2,4,3,2) -

(1,2,4) -
(1,2,4,3) -
(1,2,4,3,2) -
(4,3)
(4,3,2)

9 subarrays



Next greater element

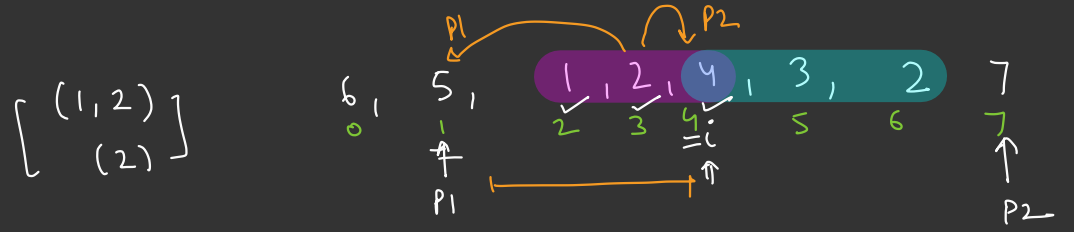
ways to choose startIdx = 3

(1, ---
2, ---
4, --

ways to choose endIdx = 3

O(1)

total subarrays with 4 as the max element = 3 × 3 = 9



O(N) == using a stack based algo

P1 = [ ] index of left greater

P2 = [ ] " " Right "

ans = 0

for (every idx → 0 to n-1) {

start$_i$ = (i - P1),

end$_i$ = (P2 - i);

ans = ans + (start$_i$ x end$_i$) x arr[idx],

4 - 1 = 3

7 - 4 = 3

contribution technique

O(N) ==

}

N = 3

P1 = [-1, -1, 1]

P2 = [1, 3, 3]

i = 0

0 - (-1) (1 - 0)

Contribution$_1$ = (1 × 1) × 1 = 1

Contribution$_4$ = $(1-(-1))^{s} \times (3-1)^{e} \times 4$

= 2 × 2 × 4

= (4) × 4

= 16

[1]

[1,4]

[1,4,3]

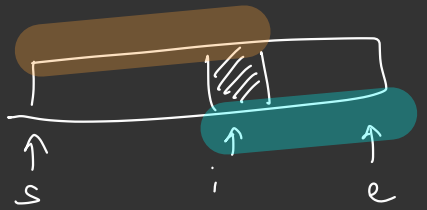
[4,3]

[4]

4 × 4

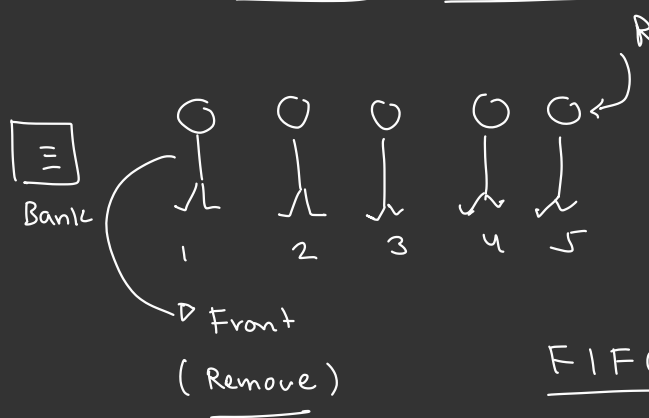
Contribution = $1 \times 1^{e} \times 3$

= 3



ans = 20

Queue (data structure)



FIFO

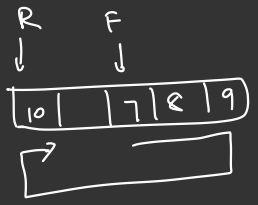
⇒ enqueue / insert / add / push / addLast → Rear end.

⇒ dequeue / remove / removeLast / pop → Front end.

⇒ peek() / front()

⇒ empty()

⇒ Queue using Array

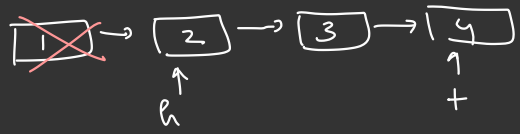


(Fixed size)

Implementation → Array
→ Linked List

⇒ Queue using Linked List (Dynamic)

Insert - Insert at tail



Remove → Remove at head

Front → ~~Ko~~ Return data from head
(peek)

empty → head is NULL

C++ Queue q = new Queue();

Java

(collection framework)

Queue <Integer> q = new LinkedList <Integer>();

(interface)

↑ java.util.LinkedList

↓ (class)

has all the functionality that a "Q class needs."

Interface

Q { add() {—} remove() {—} peek() {—}

LinkedLists

add() { ≡ }

remove() { ≡ }

[≡]

- q.add(5)
- q.remove()
- ~~poll~~ q.peek()
- q.size() == 0

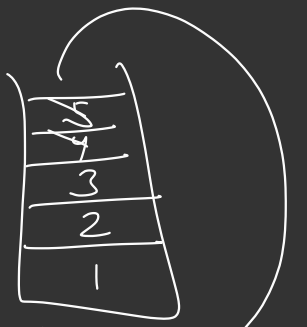
(Q) given a Queue reverse the data in the queue.

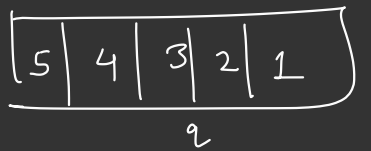
don't know anything about internals

hidden ⇒

```
Queue q = new LinkedList();

q.add(1)
q.add(2)
q.add(3)
q.add(4)
q.add(5)
```

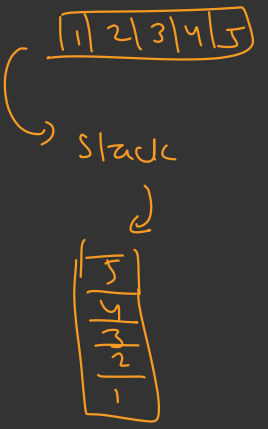




```
Stack S = new Stack();

while ( q.size() != 0 ) {
    x = q.peek()
    s.push(x)
    q.remove()
}

while ( !s.empty() ){
    x = s.top()
    q.add(x),
    s.pop();
}
```



Break

⑨ Given a Q, Reverse first k elements of same Q object

Queue q = | | | 3 | 4 | | 6 | 7 | 8 |

k = 5

q = ⇒ | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 8 |

allowed operations

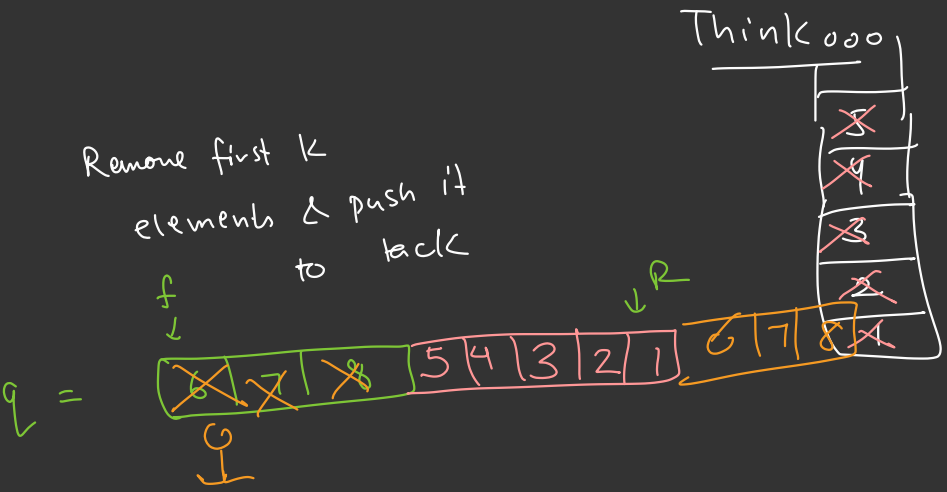
use

q [• add ( )
• remove ( )
• peek ( )]

get(i) ?

Think ooo

Remove first k elements & push it to back



HW

1) Remove K elements from the queue and add them to a stack .
2) Remove all K elements from stack and add them to the queue. (Reverse K elements)
3) Remove N-K elements from the queue and add them again to the queue.

Problem

Digits → 1, 2, 3

generate K^{th} number using above digits

⇒ 1 + 1 = 11
+ 2 = 12
+ 3 = 13

0 1 2 3 4 5 6 7 8 9 10 11

Sequence : [1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, 121, 122, ----------]

K = 11 → 33

Think - 10 40

List l,

Easy :)

l.add("1")
l add("2")
l add("3")

K=10

while (l.size() < K) {

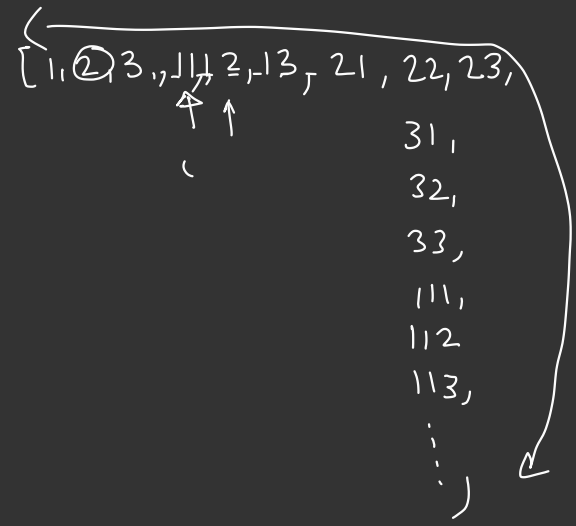
no = l[i]

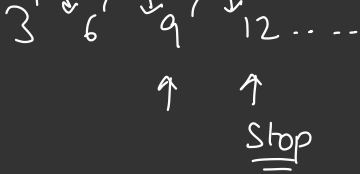
l.add(no + "1")
l add(no + "2")
l add(no + "3")

i++;

}

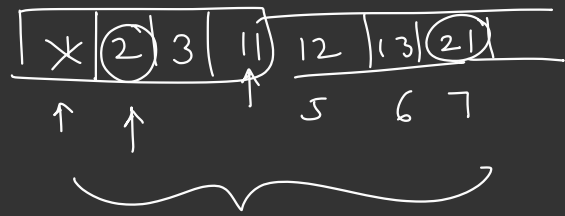
ans = l[K-1]





Queue

Queue q cnt = 7



q.add(1)
q.add(2)
q.add(3) cnt = 3

```
while ( true ){
    f = q.peek();
    q.remove()
→   q.add( f + "1" );   cnt = cnt + 1
    q.add( f + "2" );   cnt = cnt + 1
    q.add( f + "3" );
}
```

if (cnt == K) { ans = f + 1 break; } Stop

if (cnt == k) { ans = f + 2 break }

↳ cnt = cnt + 1
if (cnt == k) { } 3

→ Good Night